SIGGRAPH 2001

EXPLORE INTERACTION AND DIGITAL IMAGES

Homomorphic Factorization of BRDFs for High-Performance Rendering

Michael D. McCool

Jason Ang

Anis Ahmad

University of Waterloo

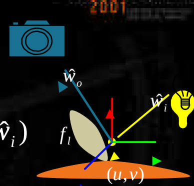
Outline

- Introduction
- Previous Work
- Factorized Representation
- Results
- Performance and Error
- Conclusions

Introduction

- What is a bidirectional reflectance distribution function (BRDF)?
- Why use BRDFs in real-time rendering?

BRDF



- Functional notation:

$$f_l(u,v,\hat{w}_o,\hat{w}_i)$$

- Assume shift-invariant:

$$f_l(\hat{w}_o,\hat{w}_i)$$

- Omit wavelength dependence (use RGB):

$$f(\hat{w}_o,\hat{w}_i)$$

BRDF

- Properties of physical BRDFs:
  - Helmholtz reciprocity
  - Conservation of energy
- BRDF classes:
  - Isotropic
  - Anisotropic

Local Lighting Equation

- **Outgoing radiance from point x in direction $\hat{w}_o$:**

$$L_o(\hat{w}_o, x) = \int_{\Omega} f(\hat{w}_o, \hat{w}_i) L_i(\hat{w}_i, x)(\hat{w}_i \bullet \hat{n}) ds(\hat{w}_i)$$

- **Illumination from N point sources:**

$$L_o(\hat{w}_o, x) = \sum_{k=1}^{N} f(\hat{w}_o, \hat{w}_i^k)(\hat{w}_i^k \bullet \hat{n}) \frac{I_k}{r_k^2}$$

Previous Work

- **Basis summation**
  - Cabral et al., Bidirectional Reflection Functions from Surface Bump Maps (1987)
  - Ward, Measuring and Modeling Anisotropic Reflection (1992)
  - Lafortune et al., Non-Linear Approximation of Reflectance Functions (1997)

Previous Work

- **Environment mapping**
  - Cabral et al., Reflection Space Image Based Rendering (1999)
  - Kautz et al., A Unified Approach to Prefiltered Environment Maps (2000)
  - Kautz and McCool, Approximation of Glossy Reflection with Prefiltered Environment Maps (2000)

Previous Work

- **Factorization**
  - Fournier, Separating Reflection Functions for Linear Radiosity (1995)
  - Heidrich and Seidel, Realistic, Hardware-Accelerated Shading and Lighting (1999)
  - Kautz and McCool, Interactive Rendering with Arbitrary BRDFs using Separable Approximations (1999)

Previous Work

- **Factorization**
  - SVD approach by Kautz and McCool (1999)

$$f(\hat{w}_o, \hat{w}_i) = \sum_{j=1}^{J} u_j(p_u(\hat{w}_o, \hat{w}_i)) \, v_j(p_v(\hat{w}_o, \hat{w}_i))$$

$$p_u : \Omega \times \Omega \rightarrow \Re^2$$

$$p_v : \Omega \times \Omega \rightarrow \Re^2$$

Homomorphic Factorization

- **Approximate f using product of positive factors:**

$$f(\hat{w}_o, \hat{w}_i) \approx \prod_{j=1}^{J} p_j(p_j(\hat{w}_o, \hat{w}_i))$$

- **Take logarithm of both sides:**

$$\tilde{f}(\hat{w}_o, \hat{w}_i) \approx \sum_{j=1}^{J} \tilde{p}_j(p_j(\hat{w}_o, \hat{w}_i))$$

Parameterization

- **Choose parameterization:**
  - Want parameters that are easy to compute
  - *Choice* (others possible!):

$$f(\hat{w}_o,\hat{w}_i) \approx p(\hat{w}_o)\, q(\hat{h})\, p(\hat{w}_i)$$

- **Take logarithm:**

$$\tilde{f}(\hat{w}_o,\hat{w}_i) \approx \tilde{p}(\hat{w}_o) + \tilde{q}(\hat{h}) + \tilde{p}(\hat{w}_i)$$

Data Constraints

- **Need to find p and q:**
  - Set up linear constraints relating samples in f to texels in p and q
  - Use bilinear weighting factors to get subpixel precision

Data Constraints

- **Data constraints can be written as:**

$$\left[\tilde{f}\right] = \left[A_p \quad A_q\right] \begin{bmatrix} \tilde{p} \\ \tilde{q} \end{bmatrix}$$

Smoothness Constraints

- **Add constraints to equate Laplacian with zero:**

$$\begin{bmatrix} \tilde{f} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} A_p & A_q \\ \lambda L_p & 0 \\ 0 & \lambda L_q \end{bmatrix} \begin{bmatrix} \tilde{p} \\ \tilde{q} \end{bmatrix}$$

  - Ensures every texel has a constraint
  - λ controls the smoothness of solution

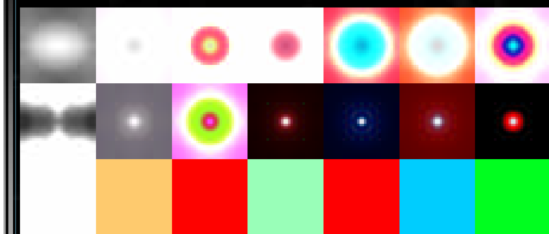
Iterative Solution

- **Solve using quasi-minimal residual (QMR) algorithm in IML++**
  - Modified conjugate-gradient algorithm
  - Freund and Nachtigal (1991)
  - Estimate an initial solution by averaging
  - Apply at sequence of increasing resolutions

Encoding into Texture Map

- **Divide p and q by their maximums and combine scale factors into a single colour** a
- **For unit-vector-valued parameters, set up texture maps as parabolic maps, hemisphere maps, or cube maps**

Results



- Top to bottom: p¢, q¢ parabolic texture maps (32 x 32) and a
- Left to right: satin (Poulin-Fournier analytic), leather, velvet (CUReT), garnet red, krylon blue, cayman, mystique (Cornell)

Rendering

- OpenGL 1.2.1 reconstruction

$$L_o(\hat{w}_o, x) =$$

$$p'(\hat{w}_o)\sum_{k=1}^{N}\left[2^s q'(\hat{h}^k)\, p'(\hat{w}_i^k)\right]\left[\frac{a\, 2^{-s} I_k (\hat{w}_i^k \bullet \hat{n})}{r_k^2}\right]$$

  - Multitexturing and compositing
  - e.g. NVIDIA GeForce 2 and ATI Radeon.

Rendering

- NVIDIA GeForce 3 reconstruction:

$$L_o(\hat{w}_o, x) =$$

$$\sum_{k=1}^{N}\left[2^s p'(\hat{w}_o) q'(\hat{h}^k)\, p'(\hat{w}_i^k)\right]\left[\frac{a\, 2^{-s} I_k (\hat{w}_i^k \bullet \hat{n})}{r_k^2}\right]$$

  - Multitexturing and compositing
  - Register combiners
  - Vertex programs

Performance

- Venus model with 90752 triangles
- Pentium III 600 MHz, 256 MB, NVIDIA GeForce 3 AGP 4x @ 1280x1024x32bit
- Standard OpenGL Lambertian lighting:
  - 123 fps, 11.2 Mtri/s
- Full illumination:
  - 76 fps, 6.9 Mtri/s

Approximation Error

Extensions

- Other parameterizations

$$f(\hat{w}_o, \hat{w}_i) \approx p(\hat{h} \cdot \hat{n}, \hat{h} \cdot \hat{w})\; q(\hat{w}_o \cdot \hat{n}, \hat{w}_i \cdot \hat{n})$$

- Material mapping

$$f(u, v, \hat{w}_o, \hat{w}_i) = \sum_{m=0}^{M} a_m(u, v)\, f_m(\hat{w}_o, \hat{w}_i)$$

Conclusions

- New BRDF factorization algorithm
  - Achieves reasonable compression ratios
  - Minimizes relative error in approximation
  - Flexible choice of parameterization
  - Results are positive factors
  - Can handle sparse data, reuse texture maps
  - Renders in real-time rates in current hardware
  - Limited to point light sources

Demo available at CAL

Acknowledgements

- Jan Kautz, Wolfgang Heidrich
- David Kirk, Matthew Papakipos, Mark Kilgard, Chris Wynn, Cass Everitt, Steve Glanville, NVIDIA
- Josée Lajoie, Kevin Moule, Martin Newell, Mira Imaging, Inc., Viewpoint Engineering
- CUReT, Cornell, Syzmon Rusinkiewicz
- NSERC, CITO